WHAT IS CLAIMED IS:

1. A method for storing and manipulating pointers in a programming language, wherein the programming language supports a native pointer type and standard pointer operations upon a native pointer object of said native pointer type, said method comprising the step of:

defining a safe pointer type supporting said standard pointer operations; and

performing automatic pointer checking in association with safe pointer type.

2. The method of claim 1 wherein said automatic pointer checking includes checking for a null pointer.

3. The method of claim 1 wherein said automatic pointer checking includes checking for improper pointer alignment.

4. The method of claim 1 further comprising a step of performing error processing.

5. The method of claim 4 wherein said error processing includes at least one of generating an error message and terminating program execution; generating a warning message without terminating program execution; and invoking a user defined error processing routine.

6. The method of claim 1 further comprising the steps of:

calling a function which returns as its value an object of said safe pointer type; and

performing said standard pointer operations upon said object.

7. The method of claim 1 further comprising the steps of:

reading a global variable of said safe pointer type; and

performing said standard pointer operation upon said object.

8. The method of claim 1 further comprising the steps of:

calling a function which returns as its value an object of said safe pointer type, said object encapsulating an improper native pointer object; and

performing error processing in response to said improper native pointer object.

9. The method of claim 1 wherein said improper native pointer object comprises one of a null pointer, misaligned pointer, and misuse of said pointer in context.

10. The method of claim 1 further comprising the steps of:

reading a global variable encapsulating an improper native pointer object; and

performing error processing in response to said improper native pointer object.

11. A safe pointer class supporting both the storage of and manipulation of a pointer in a programming language, the programming language supporting a native pointer type and standard pointer operations upon a native pointer object of said native pointer type, said safe pointer class comprising:

a safe pointer type configured to support said standard pointer operations; and

said safe pointer type further configured to support automatic pointer checking.

12. The safe pointer class of claim 11 wherein said automatic pointer checking includes checking for a null pointer.

13. The safe pointer class of claim 11 wherein said automatic pointer checking includes checking for improper pointer alignment.

14. The safe pointer class of claim 11 further comprising:

an error processing routine.

15. The safe pointer class of claim 14 wherein said error processing routine includes at least one of an error message generator and a program execution terminator; warning message generator with program execution continuer; and a user defined error processing routine.

16. The safe pointer class of claim 11 wherein said improper native pointer object comprises one of a null pointer, misaligned pointer, and misuse of said pointer in context.

17. A computer program product recorded on computer readable medium for reducing a likelihood of misuse of pointers upon which at least one software application is running, said product comprising:

computer readable means for defining a safe pointer type;

said computer readable means supporting standard pointer operations; and

said computer readable means performing automatic pointer checking.

18. The program product as claimed in claim 17 wherein said automatic pointer checking includes checking for a null pointer.

19. The program product as claimed in claim 17 wherein said automatic pointer checking includes checking for improper pointer alignment.

20. The program product as claimed in claim 17 further comprising:

computer readable means for performing error processing.